Problem: The “divide and average” method, an old-time method for approximating the square root of any positive number *a*, can be formulated as

$$x = \frac{x+\frac{a}{x}}{2} \tag{2}$$

Write a well-structured function to implement this algorithm based on the algorithm outlined in Figure 3.3.

Solution: I created a Matlab function for this problem.

Here is my function call:

>> Problem_2(1,10,.01,200) “where x = 1, a = 10, error percent desired = .01, and max iterations = 200”

Here is my answer:

X = 3.162278 EA =0.005628 Iterations =5.000000

Here is my code:

```
function Problem_2(x,a,ed,maxit)
iter = 1;
xold = -1;
ea = 100;
while  xold ~= x
    xold = x;
    x = (x+a/x)/2;
    if x ~= 0
        ea = abs((x-xold)/x)*100;
        if ea <= ed || iter >= maxit
           fprintf('X = %f EA =%f Iterations =%f \n',x,ea,iter);
           break;
        end
    end
    iter = iter+1;
end
end
```

Problem: The Maclaurin series expansion for cos x is:

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots \quad (1)$$

Starting with the simplest version, cos x = 1, add terms one at time to estimate cos(π /3). After each new term is added, compute the true and approximate percent relative errors. Use your pocket calculator to determine the true value. Add terms until the absolute value of the approximate error estimate falls below an error criterion conforming to two significant figures.

Solution: I used matlab coding in a script file to solve this problem.

Here are my answers:

Approximate Error

100	121.391441302213	9.98563898380532	0.366353197463768	0.00717361458189889

True relative Error

1	9.66227112321509	0.359240300036179	0.00708693421748174	8.66865829785013e-05

F(x) approximation

1	0.451688644383925	0.501796201500181	0.499964565328913	0.500000433432915

Here is my code:

```
%% Problem 3(book 4.1)

clear all; close all; clc;
trueval = cos(pi/3);
ed = .01;
ea = 100;
et = 1;
n = 2;
xold = 1;
x = 1;
xa(n/2)=x;
eaa(n/2) = ea;
eta(n/2) = et;
while ea>+ed
    x = x+(-1)^(n/2)*((pi/3)^n)/factorial(n);
    xa(n/2+1) = x;
    ea = abs((x-xold)/x)*100;
    eaa(n/2+1) = ea;
    et = abs((x-trueval)/(trueval))*100;
    eta(n/2+1) = et;
    n = n + 2;
    xold = x;
end
n = n/2;
fprintf('x=%u ea=%u et=%u n=%u\n',x, ea, et, n);
```

Problem: Use zero through fourth-order Taylor series expansions to predict f(2.5). for f(x) = ln(x) using a base point at x = 1. Compute the true percent relative error Et for each approximation. Discuss the meaning of the resutls

Solution: I used Matlab script files to write the code and for the Taylor series order I did hand calculations.

Here is the solution I got:
True relative error zero through 4^{th} order.
Inf 161.086048791610 344.344195166441 161.086048791610 490.950712266306

Estimates

The estimates fluctuate

0.916290731874155 -1.50000000000000 -0.375000000000000 -1.50000000000000 -0.234375000000000

This is caused by the change in sign from one order to the next and the negation of terms when the signs are opposite, but the values are the same.

Here is my code:

Problem: The Stefan-Boltzmann law can be employed to estimate the rate of radiation of energy H from a surface, as in

$$H = Ae\sigma T^4 \tag{1}$$

Where H is in watts, A = the surface area (m^2), e = the emissivity that characterizes the emitting properties of the surface (dimensionless), σ = a universal constant called the Stefan-Boltzmann constant (= 5.67×10^{-8} W $m^{-2}K^{-4}$), and T = absolute temperature (K). Determine the error of H for a copper sphere with radius 0.15 +/- 0.01 m, e = 0.90 +/- 0.05, and T 550 +/- 20. Repeat the computation but with T = 650 +/- 40.

Solution: I used Matlab script files to write the code and output the answers.

Here are my answers:

H at 550° = 1320 +/- 441, and H at 650°=2576 +/- 633

Here is my code:

```
%% Problem 5(book4.9)
clear all; close all; clc
%e = emissivity, t = absolute temp, sigma =  Boltzman constant,
e = 0.90;
eerror = .05;
t = 550;
terror = 20;
r = 0.15;
rerror = .01;
a = 4*pi*r^2;
aerror = abs(8*pi*r)*rerror;
sigma = 5.67e-8;
H = a*e*sigma*t^4;
et = abs(e*sigma*t^4)*aerror + abs(a*sigma*t^4)*eerror +
abs(a*e*sigma*4*t^3)*terror;
t2 = 650;
t2error = 40;
H2 = a*e*sigma*t2^4;
et2 = abs(e*sigma*t^4)*aerror + abs(a*sigma*t^4)*eerror +
abs(a*e*sigma*4*t^3)*t2error;
fprintf('H at 550° = %0.0f+/-%0.0f, and H at 650°=%0.0f+/-
%0.0f\n',H,et,H2,et2);
```

Problem: In a fashion similar to that in Fig 3.11, white a short program to determine the smallest number, xmin, used on the computer you will be employing with this book. Note that your computer will be unable to reliably distinguish between zero and a quantity that is smaller than this number.

Solution: I used Matlab for the solution.

The smallest number that the computer I used can interpret is: **9.881313e-324**

Here is my script:

```
%% Problem 1(book 3.4)
% Create a program that determines the smallest number, xmin, that the
computer can distinguish from zero
%Clear all previous programs, commands, and windows
    clear all; close all; clc;

    xtest = 1;
    xmin = 1;

    while xtest ~= 0
        xtest = xtest/10;
        if xtest == 0
            else
            xmin = xtest;
        end
   end

fprintf('The smallest number is: %e\n',xmin);
```